| Team Member Name | StudentID | Individual Overall Work Contribution (%) |
|---|---|---|
| Student1: | 001318881 | 25% |
| Student2: | 001294248 | 18.75% |
| Student3: | 001276650 | 18.75% |
| Student4: | 001296006 | 18.75% |
| Student5: | 001334380 | 18.75% |

# Module: COMP1821 (2023/24) / Deadline Date: 28/11/2023
# Practical group coursework

## Table of Contents

## The 5 Ps (1.a)

### ) **Problem**:

Brief Description:

The problem faced by eConsult.net is the ineffectiveness of their current system in prioritizing patient requests and the reluctance of some stakeholders, including sales representatives, towards the development of the online system. The challenges encountered by eConsult.net primarily revolve around two significant issues: the inefficiency of their existing system in handling patient requests and the resistance from certain stakeholders, notably sales representatives, towards embracing the development of an online platform.

Firstly, the inadequacy of their current system lies in its inability to effectively prioritize patient requests. This inefficiency can lead to delays in responding to critical health concerns or queries, potentially compromising patient care and satisfaction. The system might lack proper algorithms or mechanisms to identify and prioritize urgent cases, causing frustration among both patients and healthcare providers.

Secondly, the reluctance of specific stakeholders, such as sales representatives, poses a significant hurdle. This resistance could be due to several reasons, including concerns about the impact of an online system on their roles or perceived threats to existing processes that they have become accustomed to. Resistance to change is common when implementing innovative technology or systems, especially when it might disrupt established workflows or require adapting to new methods.

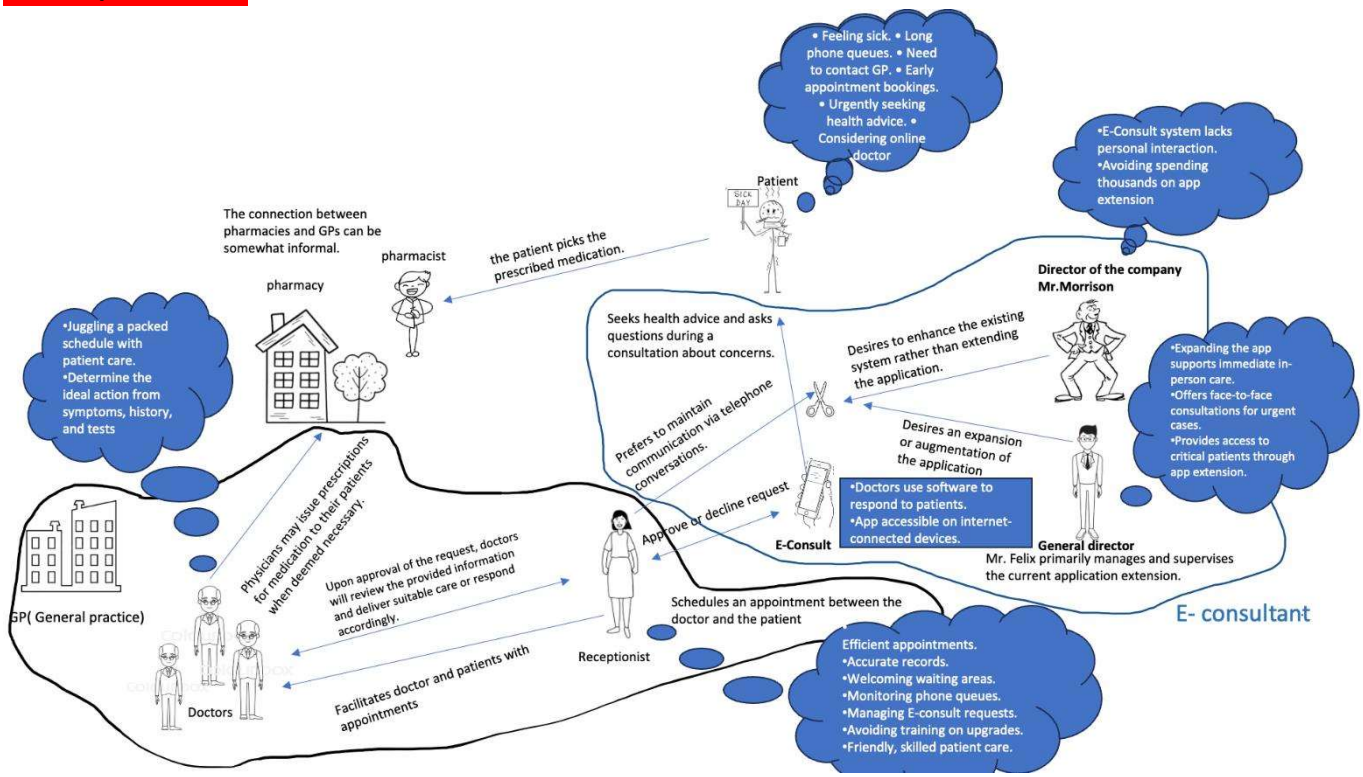
Addressing these challenges necessitates a multi-faceted approach. Improving the prioritization system requires a thorough reassessment of the current infrastructure, potentially incorporating advanced algorithms or AI-based solutions to streamline and expedite the handling of patient requests based on urgency and severity.

To tackle the resistance from stakeholders like sales representatives, effective communication and collaboration are crucial. It is essential to involve them in the decision-making process, addressing their concerns and demonstrating how the online system can complement their roles rather than replace them. Providing training and support throughout the transition can help in easing apprehensions and highlighting the benefits of the new system in enhancing overall efficiency and effectiveness.

Overall, by addressing the shortcomings of their existing system and actively engaging with stakeholders to mitigate resistance, eConsult.net can work towards implementing a

more efficient and widely accepted online platform that improves patient care and streamlines operations.

## Rich picture:




ii) Process: Waterfall Model:

Merits:

Clear Phases: The Waterfall Model follows a sequential approach with distinct phases (requirements, design, implementation, testing, deployment) making it easier to understand the project's progress and what each phase entails.

Well-defined Deliverables: Each phase produces specific deliverables, allowing for clear documentation and understanding of what needs to be accomplished at each stage.

Ease of Use: Its linear and straightforward nature makes it easy to comprehend and use for projects with well-understood requirements.

Constraints:

Limited Flexibility: Once a phase is completed and the project moves to the next stage, making changes or revisions can be challenging and costly. Requirements changes late in the process can lead to significant rework.

Agile Model:

Merits:

Flexibility: Agile is highly adaptive to changes in requirements, allowing for flexibility and accommodating modifications even in the later stages of development.

Continuous Customer Feedback: Regular iterations enable continuous feedback from stakeholders, ensuring the product aligns with evolving needs and preferences.

Iterative Development: Agile promotes iterative development cycles, allowing for quick and frequent releases of working software, enhancing adaptability and responsiveness.

Constraints:

Active Customer Involvement: Agile requires consistent and active involvement from customers or stakeholders

throughout the development process, which might not always be feasible or available.

Documentation Challenges: Due to the emphasis on quick iterations and responsiveness, Agile methodologies might sometimes lack comprehensive documentation, which can pose challenges for future reference or maintenance.

In summary, while the Waterfall Model offers a structured and well-documented approach, it can struggle with adapting to changes. On the other hand, Agile's flexibility and iterative nature are beneficial for accommodating changes but may require more consistent involvement from stakeholders and could lead to potential documentation challenges. The choice between these models often depends on the project's requirements, team dynamics, and the level of uncertainty in the project scope. Many modern approaches often blend aspects of both models to leverage their respective strengths and mitigate their limitations.

Roles in Agile Methodology:

Product Owner:

Representation of Stakeholders: The Product Owner acts as a bridge between the development team and stakeholders. They gather and prioritize requirements, ensuring that the team understands the stakeholders' needs and expectations.

Defines Requirements: The Product Owner is responsible for articulating the product vision, creating, and maintaining the

product backlog, and ensuring that the development team works on the most valuable and high-priority tasks.

Scrum Master:

Adherence to Agile Principles: The Scrum Master is a facilitator and guardian of the Agile principles and practices within the team. They coach the team on Agile methodologies, Scrum framework, and ensure adherence to its values.

Obstacle Removal: Scrum Masters focus on removing impediments that hinder the team's progress, facilitating meetings (like daily stand-ups, sprint planning, sprint review, and sprint retrospective) to ensure efficient collaboration and communication within the team.

Development Team:

Cross-functional Collaboration: The Development Team consists of individuals with varied skills necessary to deliver increments of working product. They collaborate closely to complete the work within a given iteration (sprint) and are responsible for self-organizing to achieve their goals.

Accountability for Delivering Increments: The team collectively commits to delivering working increments of the product by the end of each sprint, continuously improving their processes and productivity.

Quality Assurance:

Ensuring Product Quality: The Quality Assurance (QA) team focuses on ensuring the quality of the product through

testing and validation. They work closely with the development team to create and execute testing strategies, identify bugs, and ensure that the delivered increments meet the required quality standards.

Collaborative Effort: QA team members collaborate with developers, product owners, and other stakeholders to maintain high-quality standards throughout the development lifecycle.

Stakeholders: stakeholders in providing feedback at the end of each iteration is integral to the success of iterative development. Their insights drive continuous refinement, ensuring that the final product meets or exceeds expectations while remaining aligned with the evolving needs of the stakeholders'

Each role in Agile plays a crucial part in ensuring the success of the project. The Product Owner defines what needs to be done, the Scrum Master ensures the team follows Agile principles and removes obstacles, the Development Team works on delivering increments, and the Quality Assurance team ensures that the delivered increments meet quality standards through rigorous testing and validation. Collaboration and communication among these roles are essential for achieving the goals of Agile development.

**The Functional and Non-Functional requirements (1.b)**

functional requirement:

1. User preferences:

-users can input and save preferences like current location and preferred surgery.

2. real-time status:

-users can track the status of their request in real-time.

3. medical history access:

-users can access their previous medical history within the app.

4.priority queue:

-system prioritized patient requests based on age and health conditions.

Non-Functional Requirements:

1. Performance:

- The system should respond within 3 seconds for any user action. 2. Security:

- Patient data should be encrypted and secure. 3. Usability:

- The app should be user-friendly, requiring minimal training for both patients and sales representatives.

4. Reliability:

- The system should be available 99.9% of the time.

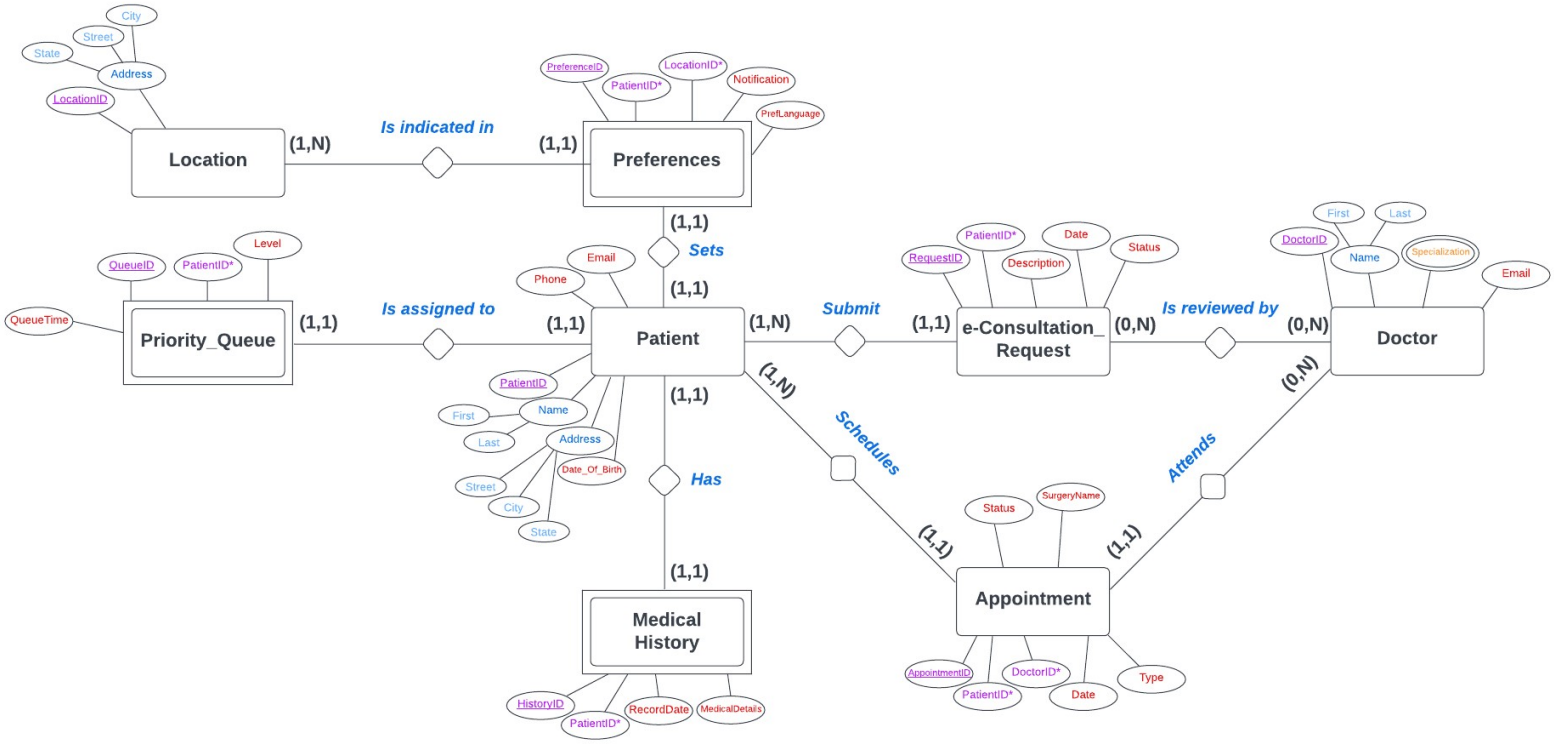
5. Scalability:

- The system should handle a growing number of users without significant performance degradation.

## The Structured Design (1.c)

Entity Relationship Diagram (I):

**Martin Style Relationship Notation.**



+

**Entities and Attributes Of ERD:**

**Patient** PatientID, Name, Address, Date_Of_Birth, Phone, Email.

Primary Key: PatientID
Foreign Key: None

**e-ConsultationRequest**(RequestID, PatientID*, Description, Date, Status)

Primary Key: RequestID
Foreign Key: PatientID* (links to entity Patient).

**Doctor**(DoctorID (Primary Key), Name, Specialization, Email)

Primary Key: DoctorID
Foreign Key: None

**Appointment**(AppointmentID (Primary Key), PatientID*, DoctorID*, Date, Type (In-person/Online), Status, SurgeryName)

Primary Key: AppointmentID
Foreign Key/s: PatientID* (links to patient), DoctorID* (links to Doctor).

**MedicalHistory**(HistoryID (Primary Key), PatientID*, RecordDate, MedicalDetails)

Primary Key: HistoryID
Foreign Key: PatientID* (links to Patient)

**Location**(LocationID (Primary Key), Address)

Primary Key: LocationID
Foreign Key: None

**PriorityQueue**(QueueID (Primary Key), PatientID*, Level, QueueTime)

Primary Key: QueueID
Foreign Key: PatientID*

**Preferences**(PreferenceID, PatientID*, LocationID*, Notification, PrefLanguage)

Primary Key: PreferenceID
Foreign Key: PatientID*, LocationID* (Links to Location)

## Relationships:

**Location and Preferences (1, N) <> (1,1):**

- Each Location "is indicated in" one or more Preferences.
- Each Preferences "indicates" one and only one Location.

**Priority Queue and Patient (1,1) <> (1,1):**

- Each Priority Queue "is assigned to" one and only one Patient.
- Each Patient "is assigned to" one and only one Priority_Queue.

**Patient and e-Consultation_Request (1, N) <> (1,1):**

- Each Patient "submits" one or more e-Consultation_Requests .
- Each e-Consultation_Request "is submitted by" one and only one Patient .

**e-Consultation_Request and Doctor (0,N) <> (0,N):**

- Each Doctor "reviews" zero or more e-Consultation_Requests .
- Each e-Consultation_Request "is reviewed by" zero or more Doctors.

**Patient and Appointment (1,N) <> (1,1):**

- Each Patient "schedules" one or more Appointments .
- Each Appointment "is scheduled by" one and only one Patient .

**Appointment and Doctor (0,N) <> (1,1):**

- Each Doctor "attends" zero or more Appointments .
- Each Appointment "is attended by" one and only one Doctor.

**Patient and Medical_History (1,N) <> (1,1):**

- Each Patient "has" one or more Medical_History.
- Each Medical_History "belongs to" one and only one Patient .

**Patient and Preferences (1,1) <> (1,1):**

- Each Patient "sets" one and only one set of Preferences .
- Each Preferences "is set by" one and only one Patient .

## Constraints:

**Location and Preferences:**

Participation is optional/partial on the part of Preferences, as a location can exist without being indicated in a preference (0,N)
Mandatory on the part of Location, as a preference must indicate one and only one location (1,1)

**Priority_Queue and Patient:**

Mandatory for both entities, as each patient must have one queue entry and each queue entry must be assigned to one and only one patient (1,1)

**Patient and e-Consultation_Request:**

Mandatory for e-Consultation_Request, as every request must be submitted by a patient (1,1)
Partial participation for Patient, as a patient may or may not submit a request (1,N)

**e-Consultation_Request and Doctor:**

Partial participation for both entities, as a request might not yet be reviewed by a doctor (0,N) and a doctor may not have reviewed any requests (0,N)

**Patient and Appointment:**

Mandatory for Appointment, as every appointment must be scheduled by a patient (1,1. Partial participation for Patient, as a patient may or may not schedule an appointment (1,N.

**Appointment and Doctor:**

Mandatory for Appointment, since every appointment must be attended by a doctor (1,1)
Partial participation for doctor, as a doctor may or may not have appointments (0,N)

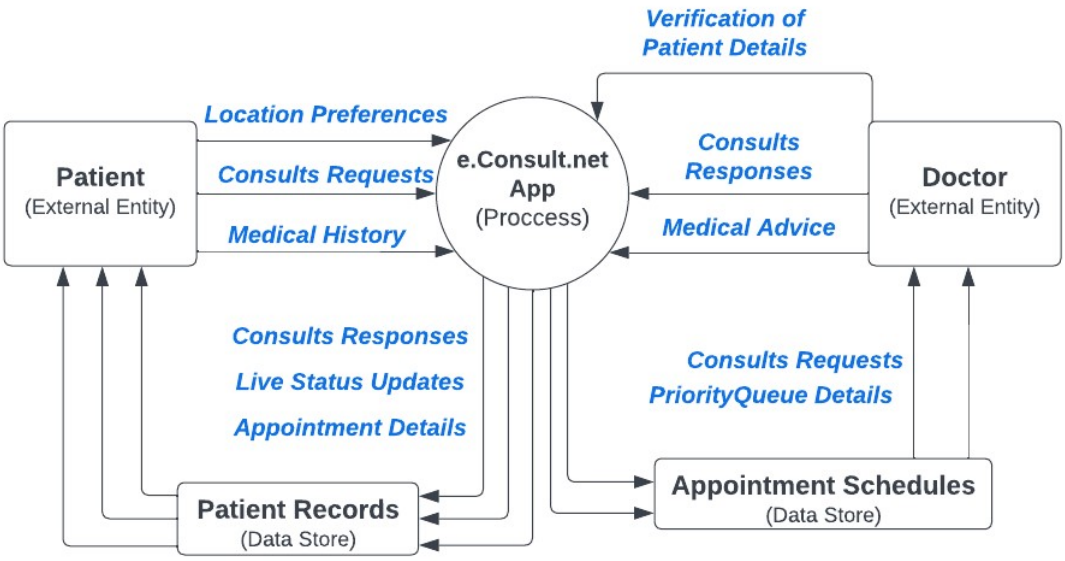
**Patient and Medical_History:**

Mandatory for Medical_History, as every medical history entry must belong to a patient (1,1)
Partial participation for patient, as a patient may or may not have a medical history entry (1,N.

**Patient and User Preferences:**

Mandatory for both entities, as each patient sets one preference, and each preference is set by one and only one patient (1,1.)

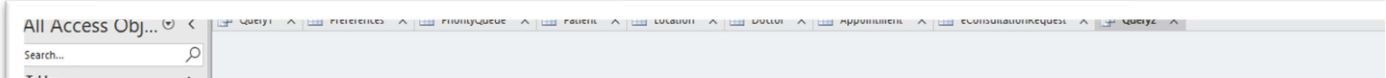
## Data Flow Diagram (ii):




**Description/Considerations of the DFD:**
The Data Flow Diagram (DFD) for e.Consult.net illustrates the main interactions between patients, doctors, and the e-Consult app.
It showcases how patients submit medical histories and consultation requests, and how they receive updates and appointment details from the e-Consult app:

- Doctors receive patient information and requests from the app and send back their medical advice.
- The app interacts with data stores for patient records and appointment schedules, indicating to the system medical data and appointment organization.
- Verification of patient details are integrated to the System app as requested to provide accurate medical advice.

## Screenshots of Database (iii): - Main View of Database Table in MF Access:



**Data Structure and Attribute Types of each Table in the Database**:

**TABLE PATIENT**

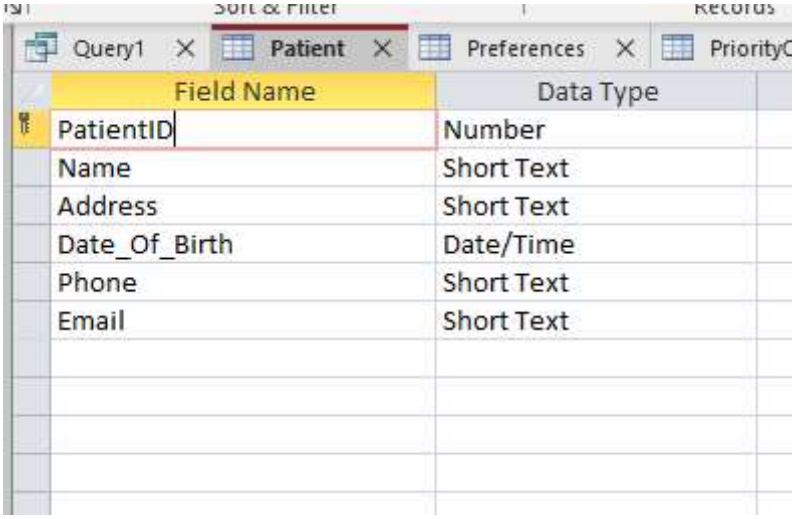



**TABLE PREFERENCES**

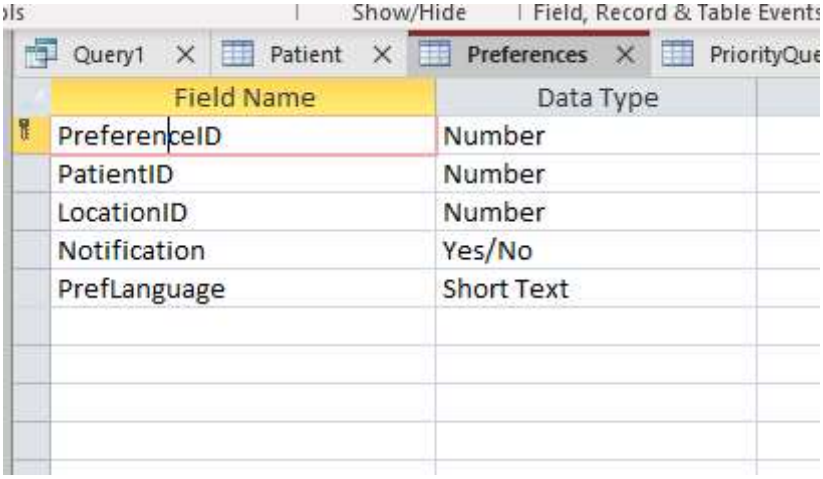



**TABLE PRIORITYQUEUE**

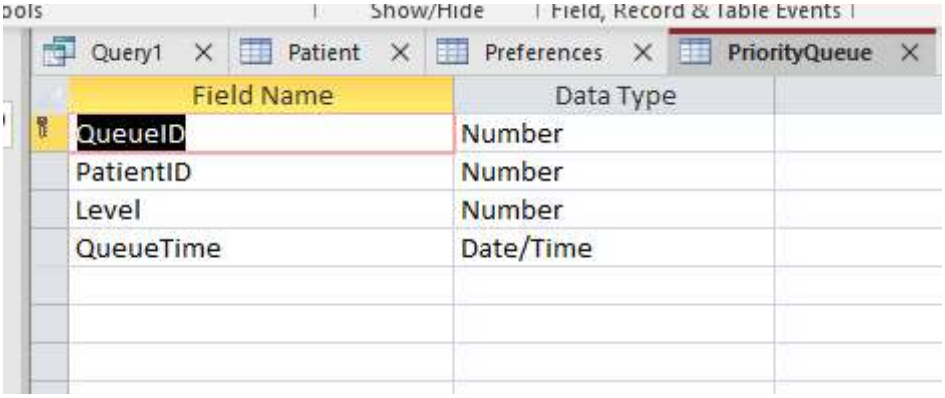



**TABLE LOCATION**

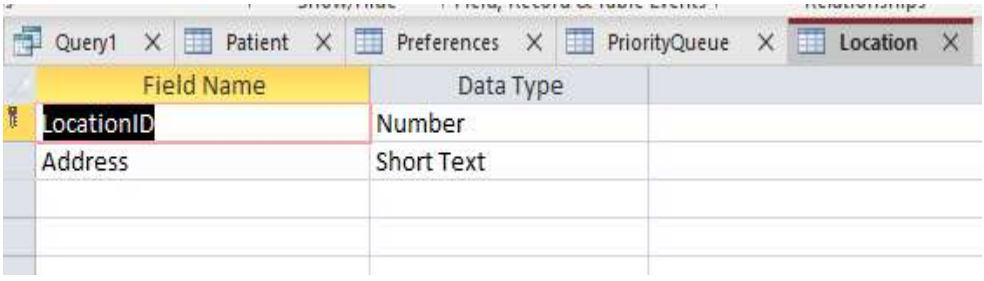



**TABLE DOCTOR**

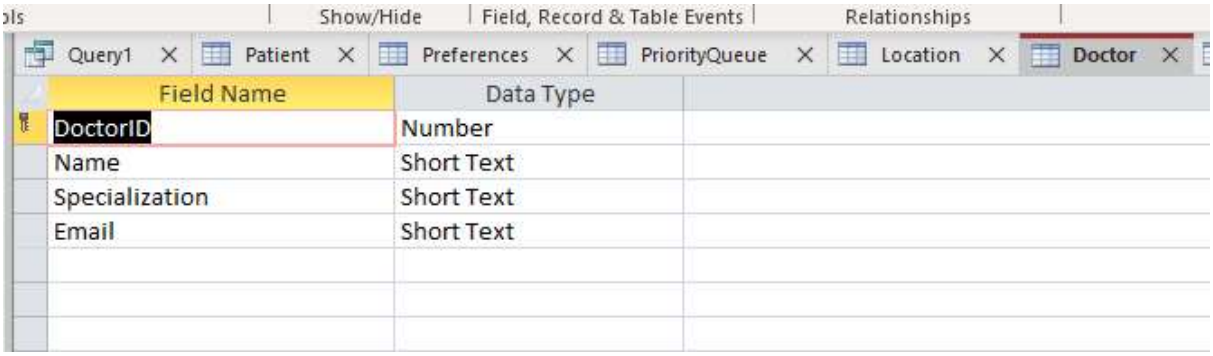



**TABLE APPOINTMENT**

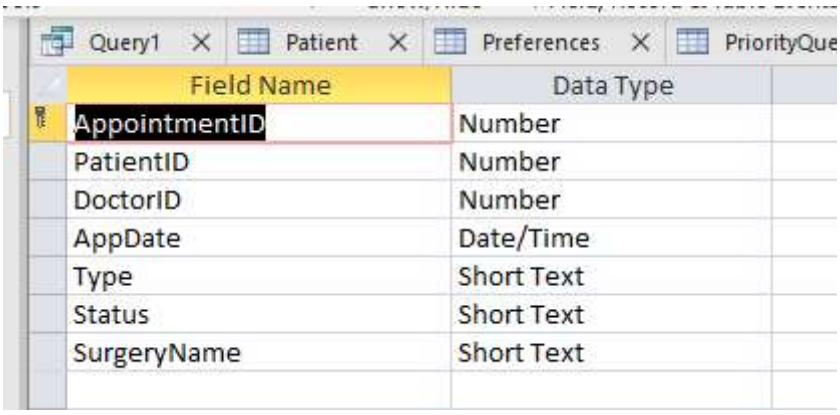
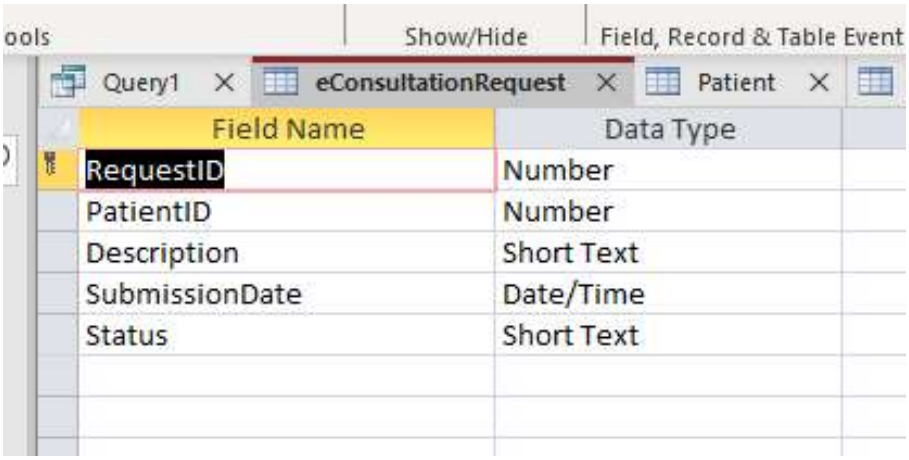



**TABLE eConsulationRequest**




Small Differences of the Nomination of Attributes in MF Access from the ERD we provided:
- Attribute “Date” in Appointment has been named AppDate to not interfere with commands in Access.
- Attribute “Notification” in Preference has been created using YESNO instead of BOOLEAN to be able to use it in MF Access.

SQL Queries used to Create the Database and his Tables in MF Access:

```
CREATE TABLE Patient (
    PatientID INT PRIMARY KEY,
```

```
    Name VARCHAR (255),
    Address VARCHAR (255),
    Date_Of_Birth DATETIME,
    Phone VARCHAR (50),
    Email VARCHAR (100)
);
CREATE TABLE Doctor (
    DoctorID INT PRIMARY KEY,
    Name VARCHAR (255),
    Specialization VARCHAR (255),
    Email VARCHAR (100)
);
CREATE TABLE Appointment (
    AppointmentID INT PRIMARY KEY,
    PatientID INT,
    DoctorID INT,
    AppDate DATETIME,
    Type VARCHAR (50),
    Status VARCHAR (50),
    SurgeryName VARCHAR (255),
    FOREIGN KEY (PatientID) REFERENCES Patient (PatientID),
    FOREIGN KEY (DoctorID) REFERENCES Doctor (DoctorID)
);
```

```
CREATE TABLE eConsultationRequest (
    RequestID INT PRIMARY KEY,
    PatientID INT,
    Description TEXT,
    SubmissionDate DATETIME,
    Status VARCHAR (50),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID));

CREATE TABLE MedicalHistory (
    HistoryID INT PRIMARY KEY,
    PatientID INT,
    RecordDate DATETIME,
    MedicalDetails TEXT,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);
CREATE TABLE Location (
    LocationID INT PRIMARY KEY,
    Address VARCHAR(255)
);
CREATE TABLE PriorityQueue (
    QueueID INT PRIMARY KEY,
    PatientID INT,
    Level INT,
    QueueTime DATETIME,
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID)
);
CREATE TABLE Preferences (
    PreferenceID INT PRIMARY KEY,
    PatientID INT,
    LocationID INT,
    Notification BOOLEAN, ( In access we used YESNO )
    PrefLanguage VARCHAR(50),
    FOREIGN KEY (PatientID) REFERENCES Patient(PatientID),
    FOREIGN KEY (LocationID) REFERENCES Location(LocationID));
```

## SQL queries:

**• Add a patient to the system:**

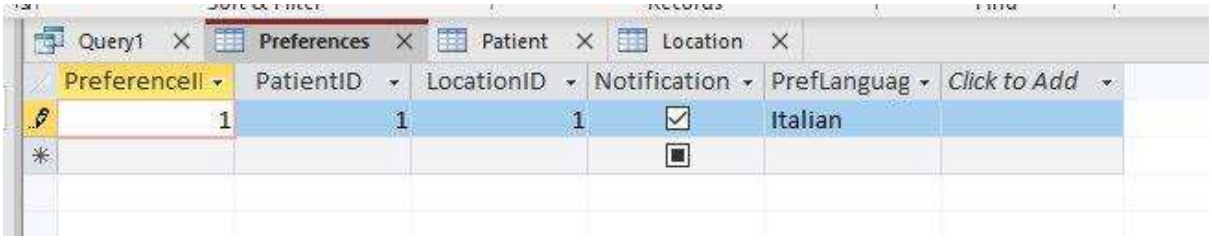
```
INSERT INTO Patient (PatientID, Name, Address, Date_Of_Birth, Phone, Email)
VALUES (1, 'Luca Marino', '255 Avery Hill', '2003-11-08', '123456789', 'lm2260r@gre.ac.uk');
```

**Result:**


**• Update patient details to inform their preferences (such as their location and nearest available GP clinic):**

```
UPDATE Preferences
SET LocationID = 1, Notification = TRUE, PrefLanguage = 'Italian'
WHERE PatientID = 1;
```

**Result:**




**• Delete patient details**

```
DELETE FROM Patient WHERE PatientID = 1;
```

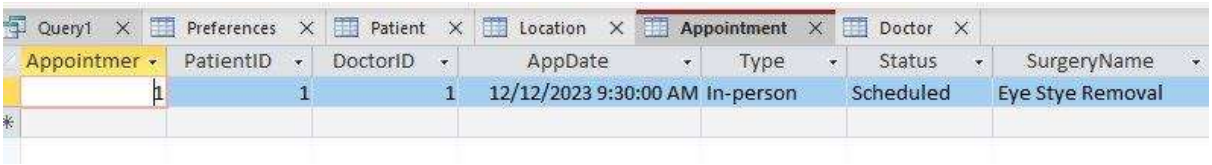
**Result:**


**Consideration:** The Alert confirms the query works since MF Access cannot delete a row when its records are being used in another table, in this case ‘Preferences’.

• **Add an appointment in the system:**

INSERT INTO Appointment (AppointmentID, PatientID, DoctorID, AppDate, Type, Status, SurgeryName)
VALUES (1, 1, 1, '2023-12-12 09:30', 'In-person', 'Scheduled', 'Eye Stye Removal');

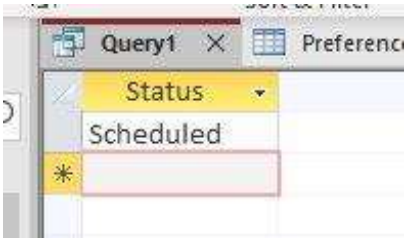
**Result:**




• **View the status of a particular appointment:**

SELECT Status FROM Appointment WHERE AppointmentID = 1;

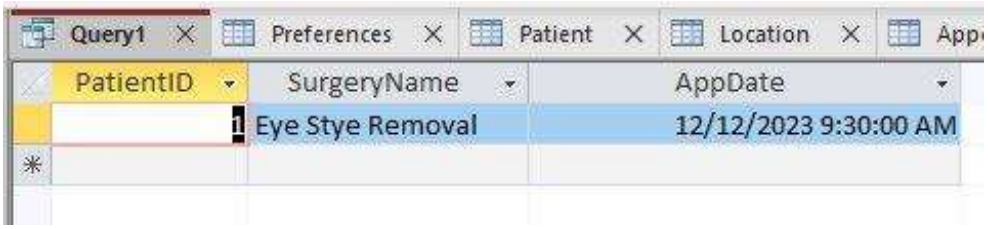
**Result:**




• **View the history of all the appointments for a particular patient including patient ID, surgery name and appointment time:**

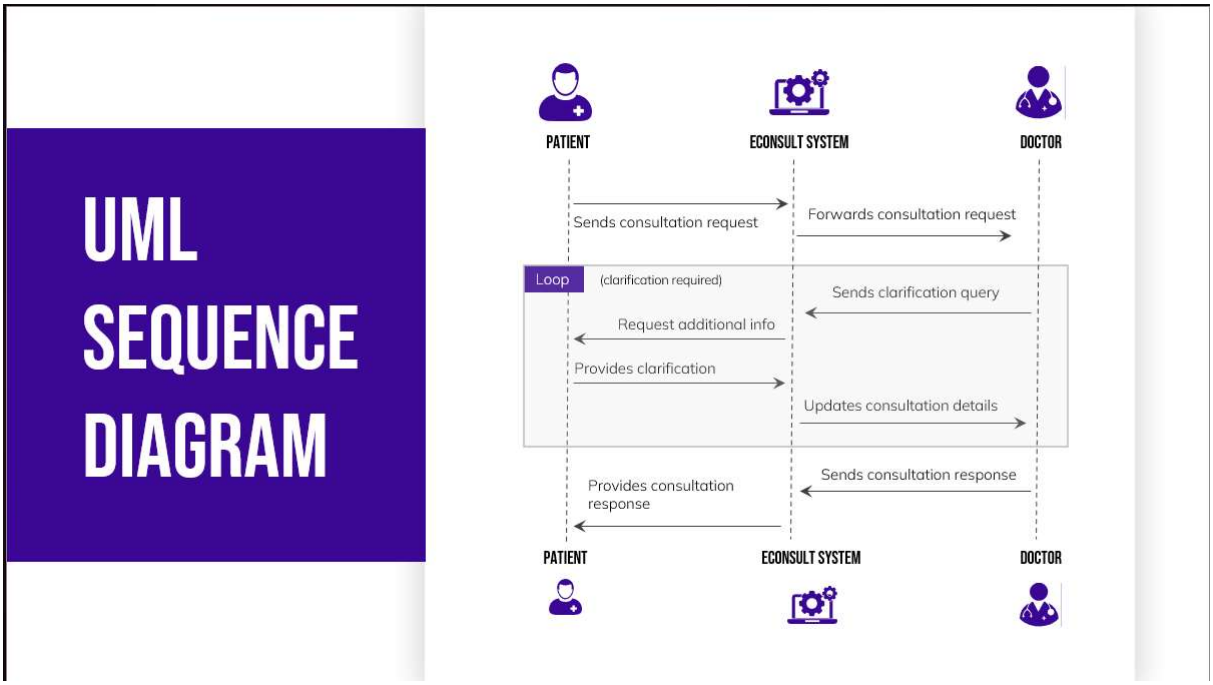
SELECT app.PatientID, app.SurgeryName, app.AppDate
FROM Appointment AS app
WHERE app.PatientID = 1;

**Result**:



## The UML Diagrams (1.d)

Use Case Diagram:



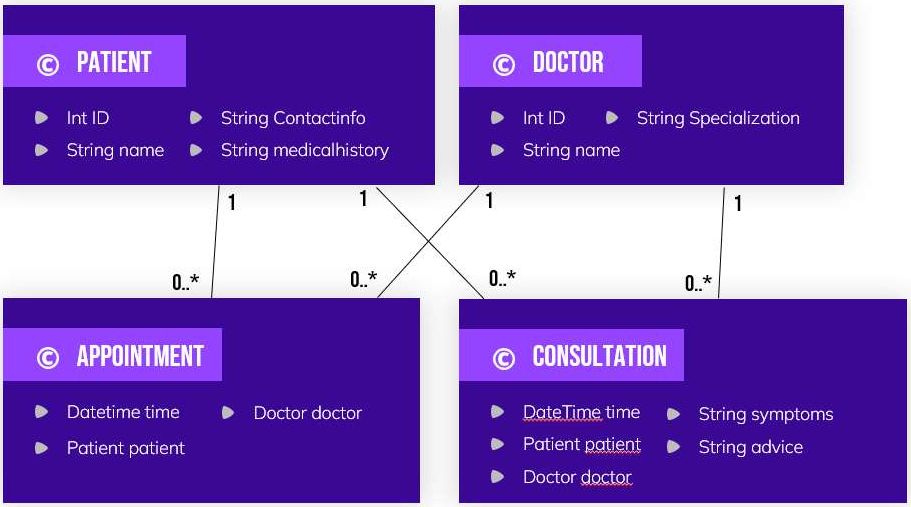


- Showing interaction between Patient, eConsult System, and Doctor
- Starting with Patient sending consultation request to system
- System forwards request to a matched Doctor
- Diagram shows a loop where Doctor asks clarification questions routed through system
- Patient provides clarification
- Finally Doctor sends consultation response via system
- System provides final response to Patient

In summary, I am showing the key steps in the workflow of a consultation involving clarification requests and responses between the Patient, System, and Doctor.

Data Class Diagram:

# DATA CLASS DIAGRAM




## CLASS DIAGRAM

- Models some of the key classes like Patient, Doctor, Appointment, and Consultation
- Shows attributes for Patient and Doctor classes - e.g. name, ID, contact info, specialization
- Models 1-to-many relationships between Patient-Appointment and Doctor-Appointment
- Also shows 1-to-many relationship between Patient-Consultation and Doctor-Consultation

Overall, it provides a simple model of some classes and relationships involved in patient appointments and consultations.

**Actors and Use Cases of the UML Use Case Diagram:**

- **Patient:** The user asking for medical advice through an e.Consult:
  - ❖ The Patient Request an e.Consult.
  - ❖ The Patient views his medical history.
  - ❖ The Patient receive a medical advice.
- **Doctor:** The person providing the medical advice and responses to the e.Consult.
  - ❖ The Doctor provides an e.Consult response
  - ❖ He prioritizes the e.Consult requests based on his medical knowledge.
- **System Administrator:** People that manage the System of appointments and preferences of the e.Consult App, maintaining the healthcare technology system. (IT)
  - ❖ The System Administrator schedule, cancel appointments and is responsible for technical updates/maintenance.
- **Healthcare Director:** decision making (Mr. Felix, Mr. Morrison) in high level decisions about system updates, analyse usage Data.
  - ❖ Analyz the entire System of the e.Consult app.

## The Self-Assessment Form (1.e)

## The Work Breakdown Form (1.f)

## References (1.h)

**In order to use the statement boolean in mf access I found the correct format type which is yes/no:** https://stackoverflow.com/questions/8940986/how-to-read-yes-no-value-as-boolean-from-an-access-database#:~:text=Boolean%20value.,NULL%20values%20are%20No%2FFalse%20.

## System Prototype Demonstration (2.a)